ALAGAPPA UNIVERSITY

[Accredited with 'A+' Grade by NAAC (CGPA:.64) in the Third Cycle and Graded as Category–I University by MHRD-UGC]

(A State University Established by the Government of Tamil Nadu)

KARAIKUDI – 630 003

Directorate of Distance Education

Master Computer Applications (M.C.A)

31544

Internet and Java Programming Lab

IV - Semester

Author

Dr. P. Prabhu

Assistant Professor in Information Technology,

Directorate of Distance Education

Alagappa University,

Karaikudi

LAB: Internet and Java Programming Lab

Syllabi

BLOCK 1: JAVA FUNDAMENTAL PROBLEMS

1 Simple Java Problems

2 Class and Objects

3 Conditional control using java

4 Looping using java

BLOCK 2: OOP CONCEPTS

5 Function overloading programs

6 Operator overloading programs

7 Inheritance programs, Packages

8 Polymorphism programs Message passing programs

BLOCK 3: THREAD & VIRTUAL FUNCTION

9 Threads

10 Virtual function programs

BLOCK 4: I/O AND EXCEPTION HANDLING

11 Exception handling programs

12 I/O manipulation programs

BLOCK 5: NETWORK PROGRAMMING

13 Applet programs

14 Implementation of simple network programs using java

Contents

Chapter 1

INTRODUCTION

1.1 Overview

Java is a programming language from Sun Microsystems (Sun) developed by James Gosling in 1991. Java's goal is to write a program once and operate it on various operating systems. New upgraded versions of Java have been published over time. Java's latest version is Java 1.8, also known as Java 8.

Java is described by a specification and consists of a programming language, a compiler, key libraries and runtime (Java virtual machine). Java runtime enables software designers to write program code in languages other than the Java programming language that still operates on the virtual Java machine. The Java platform is normally connected to the Java virtual machine and the Java core libraries. The Java Runtime Environment (JRE) and the Java Development Kit (JDK) typically come in two types. The JRE consists of the libraries of the JVM and the Java class. These have the features needed to begin Java programs.

The JDK also includes the instruments for developing Java programs. The JDK therefore consists of a Java compiler, the Java virtual machine and the Java class libraries. As plain text document, Java source files are written. Usually the programmer writes Java source code for programming in an Integrated Development Environment (IDE). An IDE promotes the programmer in writing code tasks, e.g. providing source code auto-formatting, highlighting keywords, etc.

1.2 Software Requirements

We need to install a software program called Java SE Development Kit (or short JDK, and SE means Standard Edition) to write and run a Java program. In essence, a JDK includes JRE (Java Runtime Environment): is the heart of the Java platform that allows Java programs to be run on our computer. The JRE involves JVM (Java Virtual Machine) that operates and executes Java programs by translating from bytecode to platform-dependent code (Java programs are compiled into a bytecode intermediate form) and other key libraries such as collections, File I / O, networking, etc. It also includes tools and libraries to promote the growth of Java.

Java developer should be familiar with:

- javac.exe: is Java compiler that translates programs written in Java code into bytecode form.

- java.exe: is the Java Virtual Machine launcher that executes bytecode.

1.3 Simple Example

Step 1: Open a simple text editor program such as Notepad and type the following content:

```
public class HelloWorld {
public static void main(String[] args) {
System.out.println("Hello world!");
}
}
```

Step 2: Save the file as HelloWorld.java (note that the extension is .java) under a directory, let’s say, *C:\jdk1.3\bin*

Step 3: Go to Command prompt. Type the following command to change the current directory to the one where the source file is stored:

cd C:\jdk1.3\bin

And type the following command:

javac HelloWorld.java

Step 4: Type the following command:

java HelloWorld

```
C:\jdk1.3\bin>java HelloWorld

HelloWorld!

C:\jdk1.3\bin>
```

Chapter 2

BLOCK 1: JAVA FUNDAMENTAL PROBLEMS

2.1 Simple Java Problems

This chapter covers the important concepts and principles of software development using Java. The students can able to write a computer program to solve specified problems and to use the Java SDK environment to create, debug and run simple Java programs.

Objectives:

- Understand fundamentals of programming such as variables, conditional and iterative execution, methods, etc.
- Understand fundamentals of object-oriented programming in Java, including defining classes, invoking methods, using class libraries, etc.

Scanner class in Java is found in the java.util package. Java provides various ways to read input from the keyboard, the java.util.Scanner class is one of them.

The Java Scanner class breaks the input into tokens using a delimiter which is whitespace by default. It provides many methods to read and parse various primitive values.

The Java Scanner class is widely used to parse text for strings and primitive types using a regular expression. It is the simplest way to get input in Java. By the help of Scanner in Java, we can get input from the user in primitive types such as int, long, double, byte, float, short, etc.

To get the instance of Java Scanner which reads input from the user, we need to pass the input stream (System.in) in the constructor of Scanner class. For Example:

Scanner in = **new** Scanner(System.in);

1. Scanner Class Example

```
import java.util.*;
class ScannerExample {
public static void main(String args[]){
```

```
        String s = "Welcome to Alagappa University";
        //Create scanner Object and pass string in it
        Scanner scan = new Scanner(s);
        //Check if the scanner has a token
        System.out.println("Boolean Result: " + scan.hasNext());
        //Print the string
        System.out.println("String: " +scan.nextLine());
        scan.close();
        System.out.println("--------Enter Your Details-------- ");
        Scanner in = new Scanner(System.in);
        System.out.print("Enter your name: ");
        String name = in.next();
        System.out.println("Name: " + name);
        System.out.print("Enter your age: ");
        int i = in.nextInt();
        System.out.println("Age: " + i);
        System.out.print("Enter your salary: ");
        double d = in.nextDouble();
        System.out.println("Salary: " + d);
        in.close();
        }
}
```

Output:

```
c:\jdk1.3\bin>javac ScannerExample.java
c:\jdk1.3\bin>java ScannerExample
Boolean Result: true
String: Welcome to Alagappa University.
-------Enter Your Details---------
Enter your name: Ramkumar
Name: Ramkumar
Enter your age: 23
Age: 23
Enter your salary: 25000
Salary: 25000.0
```

2. Find the Leap Year

```
class Leapyear
{
public static void main(String arg[])
{
int year=Integer.parseInt(arg[0]);
if(year!=0)
```

```
{
 if (year%400==0)
          System.out.println(year+" is a leap year");
          else  if(year%100==0)
           System.out.println(year+" is not a leap year");
           else if(year%4==0)
           System.out.println(year+" is a leap year");
           else
           System.out.println(year+" is not a leap year");
            }
           else
           System.out.println("Year zero does not exist ");
           }
}
```

Output:

```
c:\jdk1.3\bin>javac Leapyear.java
c:\jdk1.3\bin>java Leapyear 2050


2050 is not a leap year
```

3. Find the greatest number

```
import java.io.*;
class Great
{
public static void main(String v[])
{
int a,b,c;
try
{
DataInputStream d= new DataInputStream(System.in);
System.out.println("enter the value of a:");
a=Integer.parseInt(d.readLine());
System.out.println("enter the value of b:");
b=Integer.parseInt(d.readLine());
System.out.println("enter the value of c:");
c=Integer.parseInt(d.readLine());
```

```
if(a>b)
{
System.out.println("greatest no is a="+a);
}
else if(c>b)
{
System.out.println("greatest no is c="+c);
}
else
{

System.out.println("greatest no is b="+b);
}}
catch (Exception e)
{
}}}
```

Output:

```
 C:\jdk1.3\bin>javac  Great.java
C:\jdk1.3\bin>java  Great
enter value of a:
10
enter the value of b:
12
enter the value of c:
6
greatest no is b 12
```

4. Swapping Two Numbers

```
public class Swap {

public static void main(String[] args) {

                    int num1 = 10;
                    int num2 = 20;
```

```
        System.out.println("Before Swapping");
        System.out.println("Value of num1 is :" + num1);
        System.out.println("Value of num2 is :" +num2);

        //add both the numbers and assign it to first
        num1 = num1 + num2;
        num2 = num1 - num2;
        num1 = num1 - num2;

        System.out.println("Before Swapping");
        System.out.println("Value of num1 is :" + num1);
        System.out.println("Value of num2 is :" +num2);
    }
}
```

Output

```
c:\jdk1.3\bin>javac Swap.java
c:\jdk1.3\bin>java Swap

Before Swapping
Value of num1 is :10
Value of num2 is :20
Before Swapping
Value of num1 is :20
Value of num2 is :10
```

<u>5. Fibbonacci Series</u>

```
import java.io.*;
public class fibbonaci
 {
public static void main(String[] args) {
int count = 7, num1 = 0, num2 = 1;
System.out.print("Fibonacci Series of "+count+" numbers:");
for (inti = 1; i<= count; ++i)
{
System.out.print (num1+" ");
int sum = num1 + num2;
 num1 = num2;
num2 = sum;
```

```
}
}
```

Output:

```
C:\jdk1.3\bin>javac  fib.java

C:\jdk1.3\bin>java  fib

Fibonacci Series of 7 numbers: 0  1  1  2  3  5  8
```

6. Finding CGPA

```
class CGPA
{
  public static void main(String args[])
  {
     int n;
     n=args.length;
     double marks[]=new double[n];
     double grade[]=new double[n];
     double cgpa,sum=0;

    for(int i=0;i<n;i++)
     {
       marks[i]=Long.parseLong(args[i]);
     }
      for(int i=0;i<n;i++)
     {
       grade[i]=(marks[i]/10) ;
     }
      for(int i=0;i<n;i++)
     {
       sum+=grade[i];
     }
       cgpa=sum/n;
      System.out.println("cgpa="+cgpa);
      System.out.println("percantage from cgpa="+cgpa*9.5);
   }
}
```

Output:

```
c:\jdk1.3\bin> javac CGPA.java

c:\jdk1.3\bin> java CGPA 70 70 70 70 70
cgpa=7.0
percantage from cgpa=66.5
```

2.2 Class and Objects

7. Write a Java Program to define a class and its constructor

```
import java.lang.*;
class student
{
String name; int regno;
int marks1,marks2,marks3;
// null constructor
student()
{
name="raju"; regno=12345; marks1=56; marks2=47; marks3=78;
}
// parameterized constructor
student(String n, int r, int m1, int m2, int m3)
{
name = n; regno=r; marks1=m1; marks2=m2; marks3=m3;
}
// copy constructor student(student s)
{ name=s.name;
regno=s.regno;
marks1=s.marks1;
marks2=s.marks2;
marks3=s.marks3; }
void display() {
System.out.println(name + "\t" +regno+ "\t" +marks1+ "\t"
+marks2+ "\t" + marks3); }}
```

```
class studentdemo
{
public static void main(String arg[])
{
student s1=new student();
student  s2=new  student("john",34266,58,96,84);  student  s3=new
student(s1);
s1.display();
s2.display();
s3.display();
} }
```

Output:

```
c:\jdk1.3\bin>javac studentdemo.java

c:\jdk1.3\bin>java studentdemo

raju    12345   56     47     78
john    34266   58     96    84
raju    12345   56     47     78
```

8. Program to practice using String class and its methods.

```
import java.lang.String;
class stringdemo
{
public static void main(String arg[]) {
String s1=new String("alagappa university");
String s2= new String ("ALAGAPPA UNIVERSITY");

System.out.println(" The string s1 is : " +s1);
System.out.println(" The string s2 is : " +s2);
System.out.println(" Length of the string s1 is : " +s1.length());
```

```
System.out.println(" The first occurrence of u is at the position : "
+s1.indexOf('u'));
System.out.println(" The String in Upper Case : " +s1.toUpperCase());
System.out.println(" The String in Lower Case : " +s2.toLowerCase());
System.out.println(" s1 equals to s2 : " +s1.equals(s2));
System.out.println(" s1 equals ignore case to s2 : "
+s1.equalsIgnoreCase(s2));
int result=s1.compareTo(s2);
System.out.println("After compareTo()");
if(result==0)
System.out.println( s1 + " is equal to "+s2);
else if(result>0)
System.out.println( s1 + " is greather than to "+s2);
else
System.out.println( s1 + " is smaller than to "+s2);
System.out.println(" Character at an index of 6 is :" +s1.charAt(6));
String s3=s1.substring(5,8);
System.out.println(" Extracted substring is :"+s3);
System.out.println(" After Replacing a with g in s1 : " + s1.replace('a','g'));
String s4=" This is a book ";
System.out.println(" The string s4 is :"+s4);
System.out.println(" After trim()       :"+s4.trim());
}
}
```

Output:

```
c:\jdk1.3\bin>javac stringdemo.java
c:\jdk1.3\bin>java stringdemo
The string s1 is : alagappa university
The string s2 is : ALAGAPPA UNIVERSITY
Length of the string s1 is : 19
The first occurrence of u is at the position : 9
The String in Upper Case : ALAGAPPA UNIVERSITY
```

```
The String in Lower Case : alagappa university
s1 equals to s2 : false
s1 equals ignore case to s2 : true
After compareTo()
alagappa university is greather than to ALAGAPPA UNIVERSITY
Character at an index of 6 is :p
Extracted substring is :appa
After Replacing a with g in s1 : glgggppg university
The string s4 is :   This is a book
After trim        :This is a book
```

9. Program to practice using String Buffer class and its methods.

```
import java.lang.String;
class stringbufferdemo
{
public static void main(String arg[])
{
StringBuffer sb=new StringBuffer("This is my college");
System.out.println("This string sb is : " +sb);
System.out.println("The length of the string sb is : " +sb.length());
System.out.println("The capacity of the string sb is : " +sb.capacity());
System.out.println("The character at an index of 6 is : " +sb.charAt(6));
sb.setCharAt(3,'x');
System.out.println("After setting char x at position 3 : " +sb);
System.out.println("After appending : " +sb.append(" in karaikudi "));
System.out.println("After inserting : " +sb.insert(19,"ALU "));
System.out.println("After deleting : " +sb.delete(19,22));
}
}
```

Output:

```
 :\jdk1.3\bin>javac stringbufferdemo.java
c:\jdk1.3\bin>java stringbufferdemo

This string sb is : This is my college
The length of the string sb is : 18
The capacity of the string sb is : 34
```

```
The character at an index of 6 is : s
After setting char x at position 3 : Thix is my college
After appending  : Thix is my college in karaikudi
After inserting : Thix is my college ALU in karaikudi
After deleting  : Thix is my college  in karaikudi
```


10. Program to implement Vector class and its methods.

```
import java.lang.*;
import java.util.Vector;
import java.util.Enumeration;
class vectordemo
{
public static void main(String arg[])
{
Vector v=new Vector(); v.addElement("one");
v.addElement("two"); v.addElement("three");
v.insertElementAt("zero",0); v.insertElementAt("oops",3);
v.insertElementAt("four",5);
System.out.println("Vector Size :"+v.size());
System.out.println("Vector apacity :"+v.capacity());
System.out.println(" The elements of a vector are :");
Enumeration e=v.elements();
while(e.hasMoreElements())
System.out.println(e.nextElement() +" ");
System.out.println();
System.out.println("The first element is : " +v.firstElement());
System.out.println("The last element is : " +v.lastElement());
System.out.println("The object oops is found at position :
"+v.indexOf("oops"));
v.removeElement("oops");
v.removeElementAt(1);
System.out.println("After removing 2 elements ");
```

```
System.out.println("Vector Size :"+v.size());
System.out.println("The elements of vector are :");
for(int i=0;i<v.size();i++) System.out.println(v.elementAt(i)+" ");
} }
```

Output:

```
C:\jdk1.6.0_26\bin>javac vectordemo.java

C:\jdk1.6.0_26\bin>java vectordemo
Vector Size :6
Vector Capacity :10
The elements of a vector are :
zero
one
two
oops
three
four
The first element is : zero
The last element is : four
The object oops is found at position : 3
After removing 2 elements
Vector Size :4
The elements of vector are :
zero
two
three
four
```

11. Program to implement Wrapper classes and their methods.

```
import java.io.*;
class wrapperdemo {
public static void main(String args[]) {
Float P=new Float(0); Float I=new Float(0); int y=0;
try {
DataInputStream ds=new DataInputStream(System.in);
System.out.println("ENTER THE PRINCIPAL AMOUNT");
System.out.flush();
```

```
String sp=ds.readLine();
P=Float.valueOf(sp);
System.out.println("ENTER THE INTEREST RATE");
System.out.flush();
String SI=ds.readLine();
I=Float.valueOf(SI);
System.out.println("ENTER THE NUMBER OF YEARS");
System.out.flush();
String sy=ds.readLine();
y=Integer.parseInt(sy);
}
catch(Exception e)
{
System.out.println("INPUT OUTPUT ERROR");
System.exit(1);
}
float value=loan(P.floatValue(),I.floatValue(),y);

System.out.println("FINAL VALUE IS:"+value);
}
static float loan(float P,float I,int y) {
int year=1; float sum=P;
while(year<=y) {
sum=sum+(P*I)/100; year++;
}
return sum;
}}
```

Output:

```
C:\jdk1.3\bin>javac wrapperdemo.java
C:\jdk1.3\bin>java wrapperdemo
ENTER THE PRINCIPAL AMOUNT 1000
ENTER THE INTEREST RATE 2
ENTER THE NUMBER OF YEARS 1
FINAL VALUE IS:1020.0
```

```
E:\jdk1.3\bin>java wrapperdemo
ENTER THE PRINCIPAL AMOUNT 1000
ENTER THE INTEREST RATE 2
ENTER THE NUMBER OF YEARS 2
FINAL VALUE IS:1040.0
```

2.3 Conditional control in java

12. Program to implement array of objects.

```
import java.lang.*;
public class EmployeeTest {
public static void main(String[] args) {
Employee[] staff = new Employee[3];
staff[0] = new Employee("Ram", 3500);
staff[1] = new Employee("Sudha", 7500);
staff[2] = new Employee("Tony", 3800);
for (int i = 0; i < 3; i++)
staff[i].print(); }
}
class Employee {
private String name; private double salary;
public Employee(String n, double s) {
name = n; salary = s;
}
public void print() {
System.out.println(name + " " + salary); }
}
```

Output:

```
C:\jdk1.3\bin>javac EmployeeTest.java
C:\jdk1.3\bin>java EmployeeTest
Ram 3500.0
Sudha 7500.0
Tony  3800.0
```

13. Program to demonstrate this pointer

```
import java.lang.*;
class emp {
String name; int id;
String address;
void getdata(String name,int id,String address) {
this.name=name; this.id=id; this.address=address;
}
void putdata() {
System.out.println("Employee details are :");
System.out.println("Name :" +name);
System.out.println("ID :" +id);
System.out.println("Address :" +address);
} }
class empdemo {
public static void main(String arg[]) {
emp e=new emp(); e.getdata("Raja",76859,"Alagappa"); e.putdata();} }
```

Output:

```
c:\jdk1.3\bin>javac empdemo.java

c:\jdk1.3\bin>java empdemo

Employee details are :
Name     :Raja
ID       :76859
Address : Alagappa
```

14. Reverse Array

```
import java.util.Scanner;
public class reverse
{
  public static void main(String args[])
  {
```

```
        int counter, i=0, j=0, temp;
        int number[] = new int[100];
        Scanner scanner = new Scanner(System.in);
        System.out.print("How many elements you want to enter: ");
        counter = scanner.nextInt();

        for(i=0; i<counter; i++)
        {
          System.out.print("Enter Array Element"+(i+1)+": ");
          number[i] = scanner.nextInt();
        }

        j = i - 1;
        i = 0;
        scanner.close();
        while(i<j)
        {
          temp = number[i];
          number[i] = number[j];
          number[j] = temp;
          i++;
          j--;
        }
        System.out.print("Reversed array: ");
        for(i=0; i<counter; i++)
        {
          System.out.print(number[i]+ "  ");
        }
}
}
```

Output:

```
c:\jdk1.3\bin>javac reverse.java

c:\jdk1.3\bin>java reverse
How many elements you want to enter: 5
Enter Array Element1: 11
Enter Array Element2: 22
Enter Array Element3: 33
Enter Array Element4: 44
Enter Array Element5: 55
Reversed array: 55  44  33  22  11
```

2.4 Looping using java

15. Program for alphabetical order

```
import java.io.*;
class alpha
{
public static void main(String args[])throws IOException
{
String name[],t;
int i,j,x,n;
DataInputStream ds=new DataInputStream(System.in);
System.out.println("Enter the number of member");
n=Integer.parseInt(ds.readLine());
name=new String[n];
System.out.println("Enter the "+n+" name");
for(i=0;i<n;i++)
{
name[i]=ds.readLine();

}
for(i=0;i<n-1;i++)
{
for(j=0;j<n-1-i;j++)
{
x=name[j].compareTo(name[j+1]);
if(x>0)
{
t=name[j];
name[j]=name[j+1];
name[j+1]=t;
}
}}
System.out.println("ALPHABETICAL ORDER");
for(i=0;i<n;i++)
```

```
System.out.println(name[i]);
}
}
```

Output:

```
C:\JDK11~1.7B\bin>java str
Enter the number of namber
3
Enter the next name
janaki
Enter the next name
banu
Enter the next name
sara
banu
janaki
sara

C:\JDK11~1.7B\bin>
```

16. Sorting set of numbers

```
import java.io.*;
class sort
{
public static void main(String args[])throws IOException
{
int a[];
int p,n,c,t,i;
DataInputStream ds=new DataInputStream(System.in);
System.out.println("Enter the number of element");
n=Integer.parseInt(ds.readLine());
a=new int[n];
System.out.println("Enter the "+n+" number");
for(i=0;i<n;i++)
{
a[i]=Integer.parseInt(ds.readLine());
}
for(p=0;p<n-1;p++)
{
```

```
for(c=0;c<n-1-p;c++)
{
if(a[c]>a[c+1])
{
t=a[c];
a[c]=a[c+1];
a[c+1]=t;
}
}
}
for(i=0;i<n;i++)
{
System.out.println(a[i]+"\t\t\t"+a[n-1-i]);
}
}
}
```

Output:

```
C:\jdk1.3\bin>javac  sort.java
C:\jdk1.3\bin>java  sort
Enter the number of element
5
Enter the next number
8
Enter the next number
3
Enter the next number
6
Enter the next number
7
Enter the next number
1
1           8
3           7
6           6
7           3
8           1
```

17. Prime Number generation

```
import java.util.Scanner;

class PrimeNumber
{
 public static void main(String args[])
 {
      int n;
      int status = 1;
      int num = 3;
      //For capturing the value of n
      Scanner scanner = new Scanner(System.in);
      System.out.println("Enter the value of n:");
      //The entered value is stored in the var n
      n = scanner.nextInt();
      if (n >= 1)
      {
        System.out.println("First "+n+" prime numbers are:");
        //2 is a known prime number
        System.out.println(2);
      }
      for ( int i = 2 ; i <=n ;  )
      {
        for ( int j = 2 ; j <= Math.sqrt(num) ; j++ )
      {
          if ( num%j == 0 )
       {
             status = 0;
             break;
        }
        }
        if ( status != 0 )
        {
          System.out.println(num);
          i++;
        }
        status = 1;
        num++;
      }
      }
}
```

Output:

```
c:\jdk1.3\bin>javac PrimeNumber .java

c:\jdk1.3\bin>java PrimeNumber
Enter the value of n:
15
First 15 prime numbers are:
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
```

18. Student Grade System

```
import java.util.Scanner;

public class grade
{
public static void main(String args[])
{

        int marks[] = new int[6];
        int i;
        float total=0, avg;
        Scanner scanner = new Scanner(System.in);


        for(i=0; i<6; i++)
        {
          System.out.print("Enter Marks of Subject"+(i+1)+":");
          marks[i] = scanner.nextInt();
          total = total + marks[i];
        }
        scanner.close();
        //Calculating average here
        avg = total/6;
        System.out.print("The student Grade is: ");
```

```
        if(avg>=80)
        {
           System.out.print("A");
        }
        else if(avg>=60 && avg<80)
        {
          System.out.print("B");
        }

        else if(avg>=40 && avg<60)
        {
           System.out.print("C");
        }
        else
        {
           System.out.print("D");
        }
        }
}
```

Output:

```
c:\jdk1.3\bin>javac grade .java

c:\jdk1.3\bin>java grade

Enter Marks of Subject1:40
Enter Marks of Subject2:80
Enter Marks of Subject3:80
Enter Marks of Subject4:40
Enter Marks of Subject5:60
Enter Marks of Subject6:60
The student Grade is: B
```

19. EMI Calculation

```
import java.util.Scanner;
class Emi
{
   public static void main(String []args)
   {
      Scanner a = new Scanner(System.in);

      double principal, rate, time, emi;

      System.out.print("Enter principal: ");
      principal = a.nextFloat();
```

```
        System.out.print("Enter rate: ");
        rate = a.nextFloat();

        System.out.print("Enter time in year: ");
        time = a.nextFloat();

        rate=rate/(12*100);
        time=time*12;


        emi= emiCalculation(principal,rate,time);

        System.out.print("Monthly EMI is= "+emi+"\n");

    }
    static double emiCalculation(double p, double r, double t)
    {
      double e= (p*r*Math.pow(1+r,t))/(Math.pow(1+r,t)-1);
      return e;
    }}
```

Output:

```
c:\jdk1.3\bin>javac Emi .java

c:\jdk1.3\bin>java Emi
Enter Principal:10000

Enter rate: 25
Enter time in year: 1
Monthly EMI is= 950.4420326390951
```

20. Calculator

```
import java.util.Scanner;

public class calc {

public static void main(String[] args) {

        double num1, num2;
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter first number:");

        num1 = scanner.nextDouble();
        System.out.print("Enter second number:");
        num2 = scanner.nextDouble();
```

```
System.out.print("Enter an operator (+, -, *, /): ");
char operator = scanner.next().charAt(0);

scanner.close();
double output;

switch(operator)
{
   case '+':
     output = num1 + num2;
      break;

   case '-':
     output = num1 - num2;
      break;

   case '*':
     output = num1 * num2;
      break;

   case '/':
     output = num1 / num2;
      break;

   default:
      System.out.printf("You have entered wrong operator");
      return;
}

System.out.println(num1+" "+operator+" "+num2+": "+output);
}}
```

Output:

```
c:\jdk1.3\bin>javac calc .java

c:\jdk1.3\bin>java calc
Enter first number:40
Enter second number:4
Enter an operator (+, -, *, /): /
40.0 / 4.0: 10.0
```

----------Try it yourself----------

1. Write a Java program for quadratic equation
2. Write a Java program that implements Quick sort algorithm for sorting a list of names in ascending order
3. Write a Java program to transpose of given matrix
4. Write a Java program for quadratic equation
5. Write a Java program to check whether the given number is Armstrong
6. Write a Java program to count number of vowels in the string

Chapter 3

BLOCK 2: OOP CONCEPTS

Object-Oriented Programming or OOPs refers to languages that use objects in programming. Object-oriented programming aims to implement real-world entities like inheritance, hiding, polymorphism etc in programming. The main aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function.

There are four main OOP concepts in Java. These are:

- **Abstraction.** Abstraction means using simple things to represent complexity. We all know how to turn the TV on, but we don't need to know how it works in order to enjoy it. In Java, abstraction means simple things like **objects**, **classes**, and **variables** represent more complex underlying code and data. This is important because it lets avoid repeating the same work multiple times.

- **Encapsulation.** This is the practice of keeping fields within a class private, then providing access to them via public methods. It's a protective barrier that keeps the data and code safe within the class itself. This way, we can re-use objects like code components or variables without allowing open access to the data system-wide.

- **Inheritance.** This is a special feature of Object Oriented Programming in Java. It lets programmers create new classes that share some of the attributes of existing classes. This lets us build on previous work without reinventing the wheel.

- **Polymorphism.** This Java OOP concept lets programmers use the same word to mean different things in different contexts. One form of polymorphism in Java is **method overloading**. That's when different meanings are implied by the code itself. The other form is **method overriding**. That's when the different meanings are implied by the values of the supplied variables. See more on this below.

3.1 Overloading

21. Program for method overloading and dynamic method invocation

```
import java.lang.*;
class add {
void display(int a,int b) {
int c=a+b;
System.out.println("The sum of " + a + " & " + b + " is " + c); }

void display(double a,double b) {
double c=a+b;
System.out.println("The sum of " + a + " & " + b + " is " + c); }
}
class add_demo {
public static void main(String arg[]) {
add obj=new add();
obj.display(10,20);
obj.display(10.2,20.2);
} }
```

Output:

```
c:\jdk1.3\bin>javac add_demo.java

c:\jdk1.3\bin>java add_demo

The sum of 10 & 20 is 30
The sum of 10.2 & 20.2 is 30.4
```

22. Program for method overloading

```
import java.io.*;
class area
{
intx,y,z;
area()
```

```
{
x=y=z=0;
}
void area(int x1)
{ int sq=x1*x1;
System.out.println("area of square="+sq);
}
void area(int x2,int y2)
{ int rect=x2*y2;
System.out.println("area of rectangle="+rect);
}}
class overloading
{
public static void main(String args[])
{ area a=new area();
a=area(10);
a.area(10,20);
}}
```

Output:

```
C:\jdk1.3\bin>javac  overloading.java

C:\jdk1.3\bin>java  overloading
area of  square=900
area of  rectangle=1200
```

23. Program to demonstrate use of nested class.

```
import java.lang.*;
class outer
{
int m=10;
class inner
{
int n=20;
void display()
```

```
{
System.out.println("m = "+m);
System.out.println("n = "+n);
} } }

class nesteddemo {
public static void main(String arg[]) {
outer outobj=new outer();
outer.inner inobj=outobj.new inner();
inobj.display();
} }
```

Output:

```
C:\jdk1.3\bin>javac nesteddemo.java

C:\jdk1.3\bin>java nesteddemo

m  = 10
n = 20
```

24. Program to demonstrate Constructor overloading and Method overloading.

```
class RectArea {
int l,b;
RectArea() {
l=-1;
b=-1;
}
RectArea(int x,int y) {
l=x;
b=y;
}
RectArea(int x) {
l=x;
b=x;
}
int computeArea() {
return(l*b);
}
int computeArea(int a,int b) {
```

```
return(a*b);
}
int computeArea(int a) {
return(a*a);
}
}
class overloading {
public static void main(String args[]) {
int area;
RectArea r1=new RectArea(10,20);
area=r1.computeArea();
System.out.println("Using Constructor overloading");
System.out.println("Area of r1 is" + area);
RectArea r2=new RectArea(25);
area=r2.computeArea();
System.out.println("Area of r2 is" + area);
RectArea r3=new RectArea();
area=r3.computeArea(15,20);
System.out.println("Using Method overloading");
System.out.println("Area of r3 is" + area);
RectArea r4=new RectArea();
area=r4.computeArea(15,25);
System.out.println("Area of r4 is" + area);
}
}
```

Output:

```
C:\jdk1.3\bin>javac overloading.java

C:\jdk1.3\bin>java overloading

Using Constructor Overloading
Area of r1 is 200
Area of r2 is 625
Using Method Overloading
Area of r3 is 300
Area of r4 is 375
```

25. Program to implement Inner class and Access Protections.

```
class OuterClass
{
int a[]=new int[10];
int sum=0,i,k=0;
void initialize()
{
System.out.println("Inside Outer class method");
for(i=0;i<10;i++)
{
a[i]=k;
k++;
}
System.out.println("Array Elements are");
for(i=0;i<10;i++)
System.out.print(" " +a[i]);
System.out.println();
}
void show()
{
InnerClass ob=new InnerClass();
ob.cal();
}
class InnerClass
{
void cal()
{
System.out.println("Inside Inner class method");
for(i=0;i<10;i++)
sum+=a[i];
System.out.println("Sum :" +sum);
}
}
}
class outerinner
{
public static void main(String args[])
{
OuterClass ob1=new OuterClass();
ob1.initialize();
ob1.show();
}
}
```

Output:

```
C:\jdk1.3\bin>javac outerinner.java
C:\jdk1.3\bin>java outerinner
Inside Outer class method
Array elements are
10 20 30 40 50 60 70 80 90
Inside Inner class method
Sum: 450
```

3.2 Inheritance

26. Demonstrate Inheritance.

```
class SuperClass
{
int a,b;
SuperClass(int x,int y)
{
a=x;
b=y;
}
void show()
{
System.out.println("In Super Class");
System.out.println(" A and B are " + a + " " + b);
}
}
class SubClass extends SuperClass
{
int ans;
int add;
SubClass(int a,int b,int c)
{
super(a,b);
ans=c;
}
void show()
{
System.out.println("In Sub Class");
System.out.println("C value is: " + ans);
super.show();
add=a+b+ans;
```

```
System.out.println("Addition of A B and C :" + add);
}
}
class inherit{
public static void main(String args[])
{
SubClass ob=new SubClass(10,20,30);
ob.show();
}
}
```

Output:

```
C:\jdk1.3\bin>javac inherit.java
C:\jdk1.3\bin>java inherit
In sub class
C value is: 30
A and B are 10 and 20
Addition of A B and C : 60
```

27. Program to demonstrate use of sub class

```
import java.lang.*;
class parent
{
int m;
void get_m(int m)
{
this.m=m; }
void display_m()
{
System.out.println("This is from parent : m = " +m);
}
}
class child extends parent {
int n;
void get_n(int n)
{
```

NOTES

```
this.n=n; }

void display_n()
{
System.out.println("This is from child : n = " +n); }
}

class childdemo
{
public static void main(String arg[])
{
child c=new child();
c.get_m(10);
c.get_n(20);
c.display_m();
c.display_n();
} }
```

Output:

```
C:\jdk1.3\bin>javac childdemo.java

C:\jdk1.3\bin>java childdemo

This is from parent : m = 10
This is from child : n = 20
```

28. Program for banking process using class and objects

```
import java.io.*;
class acc
{
int acno,balance;
void open_act(int a,int b)
{
```

```
acno=a;
balance=b;
}
void credit(int amt)
{
balance=balance+amt;
show_balance();
}
void withdraw(int amt)
{
if (balance-amt>=100)
{
balance=balance-amt;
show_balance();
}
else
System.out.println("Sorry:No enough balance");
show_balance();
}
void show_balance()
{
System.out.println("accno:"+acno);
System.out.println("Current balcnce:"+balance);
}
}
class  bank
{
public static void main(String args[])throws IOException
{
acc a=new acc();
intano,amt,ch;
DataInputStream ds=new DataInputStream(System.in);
do
{
```

```
System.out.println("1.open new account");

System.out.println("2.credit balance in your account");
System.out.println("3.withdraw balance is your account");
System.out.println("4.Show balance");
System.out.println("Exit");
System.out.println("Enter your choice");
ch=Integer.parseInt(ds.readLine());
switch(ch)
{
case 1:
System.out.println("Enter your account number");
ano=Integer.parseInt(ds.readLine());
System.out.println("Enter the initial amount");
amt=Integer.parseInt(ds.readLine());
a.open_act(ano,amt);
break;
case 2:
System.out.println("Enter your credit amount");
amt=Integer.parseInt(ds.readLine());
a.credit(amt);
break;
case 3:
System.out.println("Enter your withdraw amount");
amt=Integer.parseInt(ds.readLine());
a.withdraw(amt);
break;
case 4:
a.show_balance();
break;
case 5:
break;
}
}
```

```
while(ch<5);
}
}
```

Output:

```
C:\jdk1.3\bin>java  bank
1.   Open new account
2.   Credit balance in your account
3.   Withdraw balance in your account
4.   Show balance
5.   Exit
Enter your choice
1
Enter the account number
2973
Enter the initial amount
20000
1.   Open new account
2.   Credit balance in your account
3.   Withdraw balance in your account
4.   Show balance
5.   Exit
Enter your choice
2
Enter your credit amount
15000
accno:2973
Current balance:35000
1.   Open new account
2.   Credit balance in your account
3.   Withdraw balance in your account
4.   Show balance
5.   Exit
Enter your choice
3
Enter your withdraw amount
10000
accno:2973
Current balance:25000
1.   Open new account
2.   Credit balance in your account
3.   Withdraw balance in your account
```

```
4.   Show balance
5.   Exit
Enter your choice
4
accno:2973
Current balance:25000
1.   Open new account
2.   Credit balance in your account
3.   Withdraw balance in your account
4.   Show balance
5.   Exit
Enter your choice
5
```

3.3 Multilevel inheritance, Interfaces and Packages:

29. Program to implement inheritance and demonstrate use of method overriding.

```
import java.lang.*;
class A
{
void display() {
System.out.println("This is from class A ");
}
}
class B extends A {
void display() {
System.out.println("This is from class B ");
}
}
class AB {
public static void main(String arg[]) {
B obj=new B(); obj.display();
}
}
```

Output:

```
C:\jdk1.3\bin>javac AB.java

C:\jdk1.3\bin>java AB

This is from class B
```

30. Program to implement inheritance by applying various access controls to its data members and methods.

```
import java.io.DataInputStream;
class Student
{
private int rollno;
private String name;
DataInputStream dis=new DataInputStream(System.in);
public void getrollno()
{
try
{

System.out.println("Enter rollno ");
rollno=Integer.parseInt(dis.readLine());
System.out.println("Enter name ");
name=dis.readLine();
}
catch(Exception e){ } }
void putrollno()
{
System.out.println("Roll No ="+rollno);
System.out.println("Name ="+name);
}
}
```

```
class Marks extends Student {
protected int m1,m2,m3;
void getmarks()
{
 try
{
System.out.println("Enter marks :");
m1=Integer.parseInt(dis.readLine());
m2=Integer.parseInt(dis.readLine());
m3=Integer.parseInt(dis.readLine());
}
catch(Exception e) { } }
void putmarks() {
System.out.println("m1="+m1);
System.out.println("m2="+m2);
System.out.println("m3="+m3);
}
}
class Result extends Marks {
private float total;
void compute_display() {
total=m1+m2+m3;
System.out.println("Total marks :" +total);
} }
class MultilevelDemo {
public static void main(String arg[])
{
Result r=new Result();
r.getrollno(); r.getmarks(); r.putrollno(); r.putmarks();
r.compute_display();
} }
```

Output:

```
C:\jdk1.3\bin>javac MultilevelDemo.java

C:\jdk1.3\bin>java MultilevelDemo
Enter rollno 12345
Enter name Abirami
Enter marks :
54
78
46
Roll No =12345
Name =Abirami
m1=54
m2=78
m3=46
Total marks :178
```

31. Program to demonstrate use of implementing interfaces.

```
import java.lang.*;
interface Area
{
final static float pi=3.14F;
float compute(float x,float y);
}
class rectangle implements Area
{
public float compute(float x,float y)
{
return(pi*x*y); }
}
class circle implements Area
{
public float compute(float x,float x)
{
return(pi*x*x);
}
```

```
}
class interfacedemo
{
public static void main(String a[])
{
rectangle rect=new rectangle(); circle cir=new circle();
Area A;
A=rect;
System.out.println("Area of rectangle="+A.compute(10,20));
A=cir;
System.out.println("Area of circle="+A.compute(30,0));
}
}
```

Output:

```
C:\jdk1.3\bin>javac interfacedemo.java

C:\jdk1.3\bin>java interfacedemo

Area of rectangle=628.0
Area of circle=2,827.43
```

32. Program to demonstrate use of extending interfaces.

```
import java.lang.*;
interface Area {
final static float pi=3.14F;
double compute(double x,double y);
}
interface display extends Area
{
void display_result(double result); }
```

```
class rectangle implements display
{
public double compute(double x, double y)
{
return(pi*x*y); }
public void display_result(double result)
{
System.out.println("The Area is :" +result); }
}

class InterfaceExtendsDemo {
public static void main(String a[])
{
rectangle rect=new rectangle();
double
result=rect.compute(10.2,12.3);
rect.display_result(result);
}
}
```

Output:

```
C:\jdk1.3\bin>javac InterfaceExtendsDemo.java
C:\jdk1.3\bin>java InterfaceExtendsDemo
The Area is : 393
```

33. Program to implement the concept of importing classes from user defined package and creating packages.

```
/*Source code of package p1 under the directory C:\jdk1.3\bin>p1\edit
Student.java */

package p1;
public class Student {
int regno;
```

```
String name;
public void getdata(int r,String s)
{
regno=r; name=s;
}
public void putdata() {
System.out.println("regno = " +regno); System.out.println("name = " +
name);
} }
/* Source code of the main function under C:\jdk1.3\bin>edit
StudentTest.java */
import p1.*;
class StudentTest {
public static void main(String arg[]) {
Student s=new Student();
s.getdata(123,"xyz");
s.putdata(); }}
```

Output:

```
C:\jdk1.3\bin>javac p1\Student.java
C:\jdk1.3\bin>javac StudentTest.java
C:\jdk1.3\bin>java StudentTest
regno = 123
name = xyz
```

34. User defined package:

```
/*Source code of package package1 under the directory
C:\jdk1.3\bin>package1\edit cal.java */
        package package1;
        public class cal
        {
```

```
public double volume(double h,double w,double d)
{
return(h*w*d);
}
public int add(int x,int y)
{
return(x+y);
}
public int divide(int x,int y)
{
return(x/y);
}
}
```

/* Source code of the main function under C:\jdk1.3\bin>edit packagedemo.java */

```
import java.io.*;

import package1.cal;

class packagedemo

{

public static void main(String args[])

{

calcalc=new cal();

int sum=calc.add(10,20);

double vol=calc.volume(10.4,13.26,32.326);

int div=calc.divide(20,4);

System.out.println("ADD"+sum);

System.out.println("VOLUME"+vol);

System.out.println("Division"+div);

}}
```

Output:

```
C:\jdk1.3\bin>javac package\cal.java

C:\jdk1.3\bin>javac packagedemo.java

C:\jdk1.3\bin>java packagedemo

ADD 30

VOLUME 4457.884704

Division 5
```

Note: Java does not support operator overloading

----------Try it yourself----------

1. Write a java program to convert the content of a given file into the uppercase content of the same file.
2. Write a Java program that displays the number of characters, lines and words in a text file.
3. Write a java program to calculate gross salary & net salary using inheritance and taking the following data. Input: empno, empname, basic (Process: DA=50%of basic HRA=25%of basic CCA=Rs240/- PF=10%of basic PT=Rs100/-)
4. Write a java program to find the details of the students eligible to enroll for the examination using interfaces

Chapter 4

BLOCK 3: THREAD

4.1 Thread:

A thread is a light-weight smallest part of a process that can run concurrently with the other parts(other threads) of the same process. Threads are independent because they all have separate path of execution that's the reason if an exception occurs in one thread, it doesn't affect the execution of other threads. All threads of a process share the common memory. **The process of executing multiple threads simultaneously is known as multithreading.**

Objectives:

- The main purpose of multithreading is to provide simultaneous execution of two or more parts of a program to maximum utilize the CPU time. A multithreaded program contains two or more parts that can run concurrently. Each such part of a program called thread.
- Threads are lightweight sub-processes, they share the common memory space. In Multithreaded environment, programs that are benefited from multithreading, utilize the maximum CPU time so that the idle time can be kept to minimum.

A thread can be in one of the following states:

- **NEW** – A thread that has not yet started is in this state.
- **RUNNABLE** – A thread executing in the Java virtual machine is in this state.
- **BLOCKED** – A thread that is blocked waiting for a monitor lock is in this state.
- **WAITING** – A thread that is waiting indefinitely for another thread to perform a particular action is in this state.
- **TIMED_WAITING** – A thread that is waiting for another thread to perform an action for up to a specified waiting time is in this state.
- **TERMINATED** – A thread that has exited is in this state. A thread can be in only one state at a given point in time.

35. Program to implement the concept of threading by extending Thread Class

```
import java.lang.Thread;
class A extends Thread {
public void run() {
System.out.println("thread A is started:");
for(int i=1;i<=5;i++)
{
System.out.println("\t from thread A:i="+i); }
System.out.println("exit from thread A:"); }
}
class B extends Thread {
public void run() {
System.out.println("thread B is started:");
for(int j=1;j<=5;j++)
{
System.out.println("\t from thread B:j="+j); }
System.out.println("exit from thread B:"); }
}
class C extends Thread {
public void run() {
System.out.println("thread C is started:");
for(int k=1;k<=5;k++)
{
System.out.println("\t from thread C:k="+k); }
System.out.println("exit from thread C:"); }
}
class Threadtest {
```

```
public static void main(String arg[]) {
new A().start();
new B().start();
new C().start();
} }
```

Output:

```
C:\jdk1.3\bin>javac Threadtest.java
C:\jdk1.3\bin>java Threadtest
thread A is started:
thread B is started:
thread C is started:
        from thread A:i=1
        from thread B:j=1
        from thread C:k=1
        from thread A:i=2
        from thread B:j=2
        from thread C:k=2
        from thread A:i=3
        from thread B:j=3
        from thread C:k=3
        from thread A:i=4
        from thread B:j=4
        from thread C:k=4
        from thread A:i=5
        from thread B:j=5
        from thread C:k=5
exit from thread A:
exit from thread B:
exit from thread
```

36. Program to implement the concept of threading by implementing Runnable Interface

```
import java.lang.Runnable;

class X implements Runnable
{
public void run()
```

```
{
for(int i=1;i<10;i++)
{
System.out.println("\t Thread X:"+i);
}
System.out.println("End of Thread X");
}
}

class Runnabletest
{
public static void main(String arg[])
{
X R=new X();
Thread T=new Thread(R);
T.start();
}
 }
```

Output:

```
C:\jdk1.3\bin>javac Runnabletest.java
C:\jdk1.3\bin>java Runnabletest
Thread X:1
Thread X:2
Thread X:3
Thread X:4
Thread X:5
Thread X:6
Thread X:7
Thread X:8
Thread X:9
End of Thread X
```

37. Program using Synchronized Threads

```
public class SynThread {
public static void main(String[] args) {
Temp c = new Temp();
Producer p1 = new Producer(c, 1);
Consumer c1 = new Consumer(c, 1);
p1.start();
c1.start();
}}

class Temp {
private int contents;
private boolean available = false;
public synchronized int get() {
while (available == false) {
try {
wait();
} catch (InterruptedException e) { }
}
available = false;
notifyAll();
return contents;
}
public synchronized void put(int value) {
while (available == true) {
try {
wait();
} catch (InterruptedException e) { }
}
contents = value;
available = true;
notifyAll();
}}

class Consumer extends Thread {
private Temp t;
private int number;
public Consumer(Temp tmp, int number) {
t = tmp;
this.number = number;
}
public void run() {
int value = 0;
```

```
for (int i = 0; i < 10; i++) {
value = t.get();
System.out.println("Consumer #" + this.number + " got: " + value);
}}}

class Producer extends Thread {
private Temp t;
private int number;
public Producer(Temp tmp, int number) {
t = tmp;
this.number = number;
}
public void run() {
for (int i = 0; i < 10; i++) {
t.put(i);
System.out.println("Producer #" + this.number + " put: " + i);
try {
sleep((int)(Math.random() * 100));
} catch (InterruptedException e) { }
}}}
```

Output:

```
C:\jdk1.3\bin>javac  SynThread.java
C:\jdk1.3\bin>java SynThread
Producer #1 put : 0
Consumer #1 got : 0
Producer #1 put : 1
Consumer #1 got : 1
Producer #1 put : 2
Consumer #1 got : 2
Producer #1 put : 3
Consumer #1 got : 3
Producer #1 put : 4
Consumer #1 got : 4
Producer #1 put : 5
Consumer #1 got : 5
Producer #1 put :6
Consumer #1 got : 6
Producer #1 put : 7
Consumer #1 got : 7
Producer #1 put : 8
Consumer #1 got : 8
```

```
Producer #1 put : 9
Consumer #1 got : 9
```

<u>Note: Java does not support virtual function concepts</u>

----------Try it yourself----------

1. Write a Java program that implements a multi-threaded program has three threads. First thread generates a random integer every 1 second and if the value is even, second thread computes the square of the number and prints. If the value is odd the third thread will print the value of cube of the number

2. Write a Java program that creates threads by extending Thread class. First thread display “Good Morning “every 1 sec, the second thread displays “Hello “every 2 seconds and the third display “Welcome” every 3 seconds ,(Repeat the same by implementing Runnable)

3. Write a Java program to implement serialization concept

Chapter 5

BLOCK 4: I/O AND EXCEPTION HANDLING

5.1. Input Output Operations:

- The java.io package contains many classes that allow us to define various streams with particular characteristics
- Some classes assume that the data consists of characters
- Others assume that the data consists of raw bytes of binary information
- Streams can be further subdivided as follows:
  - *data stream*, which acts as either a source or destination
  - *processing stream*, which alters or manipulates the basic data in the stream
- The InputStream and OutputStream classes (and their descendants) represent byte streams
- The Reader and Writer classes (and their descendants) represent character streams
- Information can be read from and written to text files by declaring and using the correct I/O streams
- The FileReader class represents an input file containing character data
- The FileReader and BufferedReader classes together create a convenient text file output stream

38. Program to demonstrate I/O operations

```
import java.io.*;
class file
{
public static void main(String args[]) throws IOException
{
int empno,salary;
String name, design;
int more;
FileOutputStream fos=new FileOutputStream("emp.java");
PrintWriter write=new PrintWriter(fos);
DataInputStream ds=new DataInputStream(System.in);
do
{
System.out.println("Enter employee no:");
empno=Integer.parseInt(ds.readLine());
System.out.println("Enter employee name:");
name=(ds.readLine());
System.out.println("Enter employee salary no:");
salary=Integer.parseInt(ds.readLine());
System.out.println("Enter designation:");
design=(ds.readLine());
write.println(empno+"\t" +name+ "\t" +design+"\t"+salary);
System.out.println("add more records=1,exit=0");
more=Integer.parseInt(ds.readLine());
}
while(more==1);
write.close();
}
}
```

Output:

```
Enter employee no:
101
Enter employee name:
Kalai
Enter employee salary
20000
Enter designation
Manager
Add more records=1, exit =0
1
Enter employee no:
102
Enter employee name:
raja
Enter employee salary
25000
Enter designation
Accounts officer
Add more records=1, exit =0
0
```

39. Program to demonstrate file read operation:

```
import java.io.*;
classfileread
{
public static void main(String args[])throws IOException
{
int empno,salary;
String name,design;
FileInputStreamfis=new FileInputStream("emp.java");
InputStreamReaderisn=new InputStreamReader(fis);
StreamTokenizer tokens=new StreamTokenizer(isn);
while(tokens.nextToken()!=tokens.TT_EOF)
{
empno=(int)tokens.nval;
tokens.nextToken();
name=tokens.sval;
```

```
tokens.nextToken();
salary=(int)tokens.nval;
design=tokens.sval;
tokens.nextToken();
salary=(int)tokens.nval;
System.out.println(empno+" "+name+" "+salary+" "+design);
}
}
}
```

Output:

```
C:\jdk1.3\bin>javac fileread.java

C:\jdk1.3\bin>java fileread
101 Kalai  20000 Manager
102 Raja    25000  Accounts officer
```

40. Program which writes a object to a file

```
import java.io.*;
class Person implements Serializable {
private String firstName;
private String lastName;
private transient String password;
public Person() {
}
public String getFirstName() {
return firstName;
}
public void setFirstName(String firstName) {
this.firstName = firstName;
}
public String getLastName() {
return lastName;
}
public void setLastName(String lastName) {
this.lastName = lastName;
}
public String getPassword() {
```

```
return password;
}
public void setPassword(String pass) {
password=pass;
}
public String toString() {

StringBuffer buffer = new StringBuffer();
buffer.append(firstName);
buffer.append("\n");
buffer.append(lastName);
buffer.append("\n");
buffer.append(password);
buffer.append("\n");
return buffer.toString();
}
}
public class filepgm {
public void writePersons(String filename) {
ObjectOutputStream outputStream = null;
ObjectInputStream ois=null;
try {
outputStream = new ObjectOutputStream(new
FileOutputStream(filename));
Person person = new Person();
person.setFirstName("Ram");
person.setLastName("Kumar");
person.setPassword("aaa");
outputStream.writeObject(person);
outputStream.flush();
person = new Person();
person.setFirstName("Gopal");
person.setLastName("Krishna");
person.setPassword("ccc");
outputStream.writeObject(person);
} catch (FileNotFoundException ex) {
ex.printStackTrace();
} catch (IOException ex) {
ex.printStackTrace();
} finally {
//Close the ObjectOutputStream
try {
```

```
if (outputStream != null) {
outputStream.flush();
outputStream.close();
}
} catch (IOException ex) {
ex.printStackTrace();
}
}
try {
Person p,q;
ois=new ObjectInputStream(new FileInputStream(filename));
p=(Person)ois.readObject();
System.out.println("Object read :" +p);
q=(Person)ois.readObject();
System.out.println("Object read :" +q);
ois.close();
}catch(Exception e)
{
e.printStackTrace();
}
}
public static void main(String[] args) {
new filepgm().writePersons("abc.txt");
}}
```

<u>Output:</u>

```
C:\jdk1.3\bin>javac  filepgm.java
C:\jdk1.3\bin>java filepgm


Object read:

Ram
Kumar
null
Object read:
Gopal
Krishna
null
```

5.2 Exception Handling:

- An *exception* is an object that describes an unusual or erroneous situation
- Exceptions are *thrown* by a program, and may be *caught* and *handled* by another part of the program
- A program can be separated into a normal execution flow and an *exception execution flow*
- An *error* is also represented as an object in Java, but usually represents a unrecoverable situation and should not be caught
- Java has a predefined set of exceptions and errors that can occur during execution
- A program can deal with an exception in one of three ways:
    - ignore it
    - handle it where it occurs
    - handle it an another place in the program
- The manner in which an exception is processed is an important design consideration. To process an exception when it occurs, the line that throws the exception is executed within a *try block*
- A try block is followed by one or more *catch* clauses, which contain code to process an exception
- Each catch clause has an associated exception type and is called an *exception handler*

41. Program to demonstrate Exception Handling (Using Nested try catch and finally).

```
class excep {
static void nestedTry(int a)
{
int sum;
try
{
if(a==1)
a=a/(a-a);
if(a==2)
{
int x[]={2,9};
x[5]=99;
}
}
catch(ArrayIndexOutOfBoundsException e)
{
System.out.println("Array index out of bounds" +e);
}
}


public static void main(String args[]) {
try
{
int a=args.length;
int b=55/a;
System.out.println("No of arguments are = " +a);
nestedTry(a);
}
catch(ArithmeticException e)
{
System.out.println("Divide by zero error" +e);
}}}
```

Output :

```
C:\jdk1.3\bin>javac excep.java

 C:\jdk1.3\bin>java excep 10
No of arguments are = 1
Divide by zero error java.lang.ArithmeticException: / by zero
```

```
C:\jdk1.3\bin>java 10 20
No of arguments are = 2
Divide by zero error java.lang.ArrayIndexOutOfBoundsException: 5
C:\jdk1.3\bin>java 10 20 30
No of arguments are =
```

42. Program to implement the concept of Exception Handling using redefined exception.

```
import java.lang.*;
class Exception_handle
{
public static void main(String argv[]) {
int a=10,b=5,c=5,x,y;
try
{
x=a/(b-c);
}
catch(ArithmeticException e)
{
System.out.println("DIVISION BY ZERO");
}
y=a/(b+c);
 System.out.println("y="+y);
} }
```

Output:

```
C:\jdk1.3\bin>javac Exception_handle.java
C:\jdk1.3\bin>java Exception_handle

DIVISION BY ZERO y=1
```

43. Program to implement the concept of Exception Handling.

```
import java.io.*;
import java.lang.*;
import java.util.*;
class except
{
public static void main(String args[])throws IOException
{
int ch;
DataInputStream ds=new DataInputStream(System.in);
do
{
System.out.println("menu");
System.out.println("1.add");
System.out.println("2.str");
System.out.println("3.array");
System.out.println("4.exit");
System.out.println("Enter your choice");
ch=Integer.parseInt(ds.readLine());
switch(ch)
{
case 1:
int a=5,b=0,c;
try
{
c=a/b;
System.out.println("Result"+c);
}
catch(Exception e)
{
System.out.println(e.toString());
e.printStackTrace();
}
```

```
break;
case 2:
String s="abcds";
try
{
System.out.println(s.charAt(5));
}
catch(Exception e)
{
System.out.println(e.toString());
e.printStackTrace();
}
break;
case 3:
int x[]=new int[5];
try
{
System.out.println(x[5]);
}
catch(Exception e)
{
System.out.println(e.toString());
e.printStackTrace();
}
break;
case 4:
break;
}
}
while(ch<5);
}
}
```

Output:

```
C:\jdk1.3\bin>javac except.java
C:\jdk1.3\bin>java except
Menu
1.add
2.str
3.array
4.exit
Enter your choice
1
java.lang.ArithmeticException: / by zero
java.lang.ArithmeticException: / by zero
                at except.main(except.java:25)
Menu
1.add
2.str
3.array
4.exit
Enter your choice
2
java.lang.StringIndexOutOfBoundsException: String index out of
range: 5
java.lang.StringIndexOutOfBoundsException: String index out of
range: 5
                at java.lang.String.charAt(String.java.658)
                at except.main(except.java:38)
Menu
1.add
2.str
3.array
4.exit
Enter your choice
3
java.lang.ArrayIndexOutOfBoundsException
java.lang.ArrayIndexOutOfBoundsException
               at except.main(Compiled Code)
Menu
1.add
2.str
3.array
4.exit
Enter your choice
4
```

44. Program to implement the concept of Exception Handling by creating user defined exceptions.

```
import java.lang.Exception;
import java.lang.*;
import java.lang.Exception;
import java.io.DataInputStream;

class MyException extends Exception {
MyException(String message)
{
super(message); }
}
class userdef {
public static void main(String a[])
{
int age;
DataInputStream ds=new DataInputStream(System.in);
try
{
System.out.println("Enter the age (above 15 and below 25) :");
age=Integer.parseInt(ds.readLine());
if(age<15 || age> 25)
{
throw new MyException("Number not in range"); }
System.out.println(" the number is :" +age); }
catch(MyException e)
{
System.out.println("Caught MyException");
System.out.println(e.getMessage());
}
catch(Exception e)
{ System.out.println(e); }
} }
```

Output:

```
:\jdk1.3\bin>java userdef

Enter the age (above 15 and below 25) : 6
Caught MyException Number not in range

c:\jdk1.3\bin>java userdef

Enter the age (above 15 and below 25) : 20
the number is :20
```

----------Try it yourself----------

1. Write a Java program that reads a file name from the user, and then displays information about whether the file exists, whether the file is readable,
2. Write a java program that loads names and phone numbers from a text file where the data is organized as one line per record and each field in a record are separated by tab (\t).
3. Write a Java program that reads on file name from the user, displays information about whether the file exists, whether the file is readable, whether the file is writable, the type of file and the length of the file in bytes?
4. Write a JAVA program for creation of illustrating throw
5. Write a JAVA program for creation of Illustrating finally

Chapter 6

BLOCK 5: NETWORK PROGRAMMING

6.1 Applet Programs:

An applet is a Java program that can be embedded into a web page. It runs inside the web browser and works at client side. An applet is embedded in an HTML page using the APPLET tag and hosted on a web server.

Applets are used to make the web site more dynamic and entertaining.

Important points:

1. All applets are sub-classes (either directly or indirectly) of java.applet.Applet class.
2. Applets are not stand-alone programs. Instead, they run within either a web browser or an applet viewer. JDK provides a standard applet viewer tool called applet viewer.
3. In general, execution of an applet does not begin at main() method.
4. Output of an applet window is not performed by *System.out.println()*. Rather it is handled with various AWT methods, such as *drawString()*.

45. Program to demonstrate Keyboard event

```
import java.applet.*;
import java.awt.event.*;
import java.awt.*;

/* <applet code="KeyEvents.class" width=300 height=200> </applet> */

public class KeyEvents extends Applet implements KeyListener {
String msg =" ";
int x=10,y=20;
public void init()
{
```

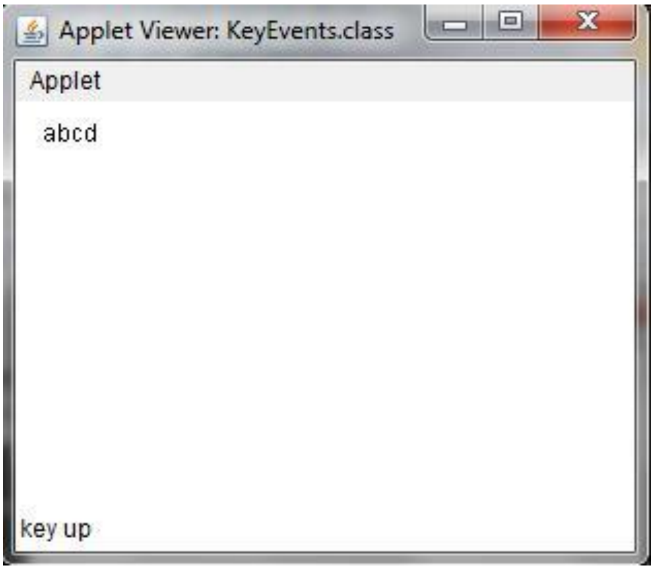
```
addKeyListener(this);
requestFocus();
}

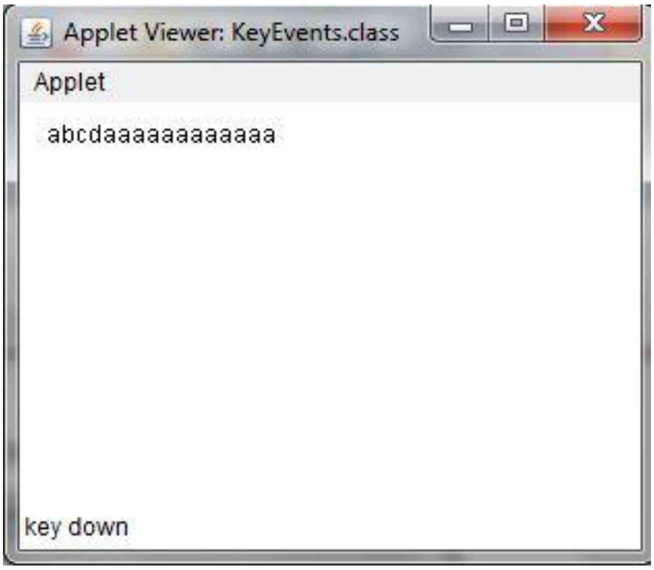
public void keyPressed(KeyEvent k)
{
showStatus("key down"); }

public void keyReleased(KeyEvent k)
{
showStatus("key up"); }
public void keyTyped(KeyEvent k)
{
msg +=k.getKeyChar();
repaint();
}
public void paint(Graphics g)
{
g.drawString(msg,x,y); }
}
```

Output

```
C:\jdk1.3\bin>javac Keyevents.java
C:\jdk1.3\bin>appletviewer Keyevents.java
```





<u>46. Program to demonstrate Mouse events</u>

```
import java.applet.*;
import java.awt.event.*;
import java.awt.*;

/* <applet code="MouseEvents.class" width=300 height=200> </applet>
*/
public class MouseEvents extends Applet implements MouseListener,
MouseMotionListener
{
```

```
String msg =" ";
int x=0,y=0;
public void init()
{
addMouseListener(this);
addMouseMotionListener(this);
}

public void mouseClicked(MouseEvent m)
{
x=10; y=10;
msg ="mouse clicked";
repaint();
}

public void mouseEntered(MouseEvent m) {
x=10; y=10;
msg ="mouse Entered";
repaint();
}

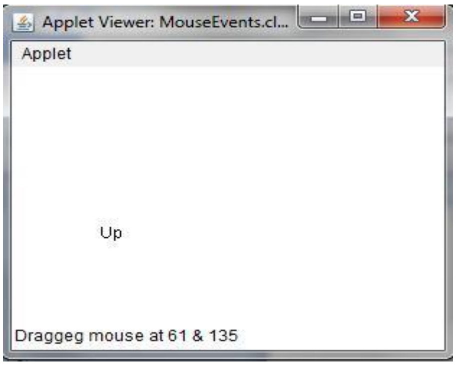
public void mouseExited(MouseEvent m) {
x=10; y=10;
msg ="mouse Exited";
repaint();
}

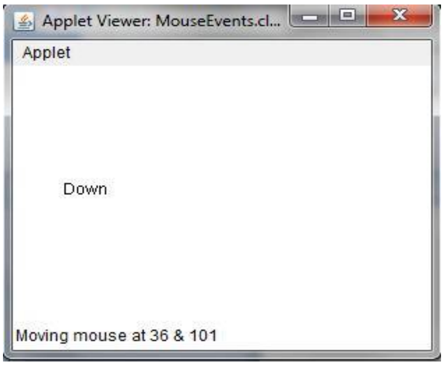
public void mousePressed(MouseEvent m) {
x=m.getX();
y=m.getY();
msg ="Down";
repaint();
}
```

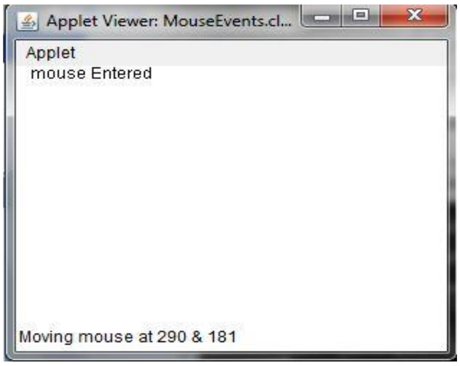
```
public void mouseReleased(MouseEvent m) {
x=m.getX();
y=m.getY();
msg ="Up";
repaint();
}
public void mouseDragged(MouseEvent m) {
x=m.getX();
y=m.getY();
msg ="*";
showStatus("Draggeg mouse at " +x+ " & "+y);
repaint();
}
public void mouseMoved(MouseEvent m) {
showStatus("Moving mouse at " +m.getX()+ " & "+m.getY()); }
public void paint(Graphics g) {
g.drawString(msg,x,y); }
}
```

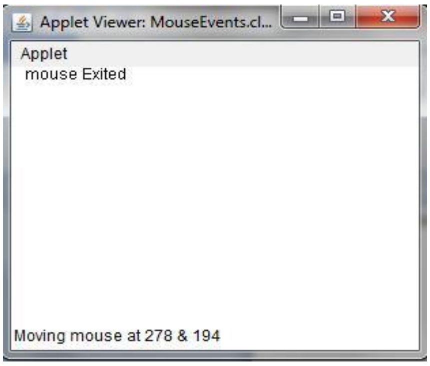
Output:

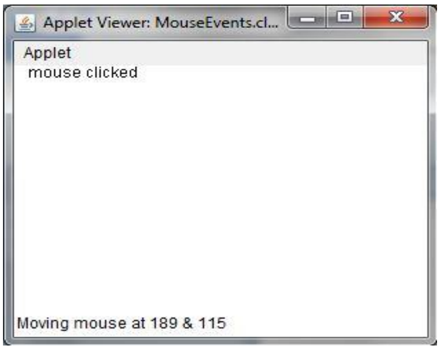
```
C:\jdk1.3\bin>javac MouseEvents.java
C:\jdk1.3\bin>appletviewer MouseEvents.java
```

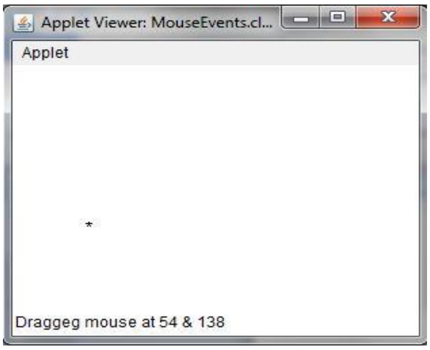












47. Write program for using Graphics class

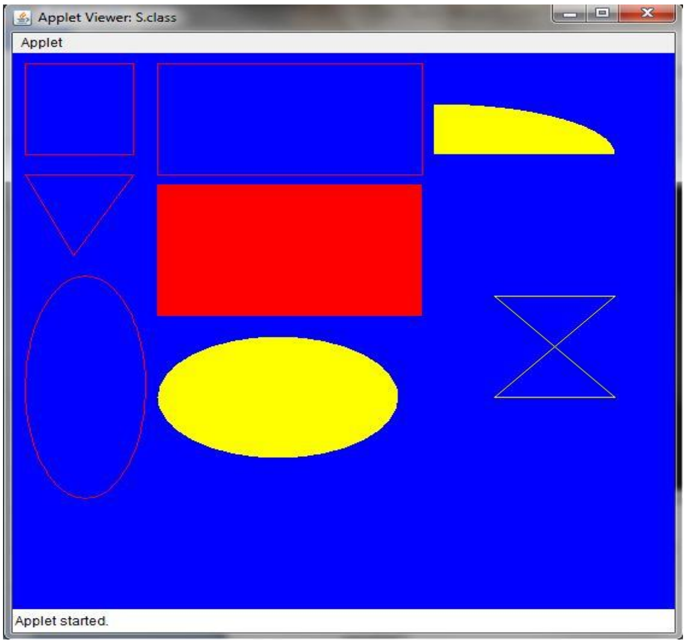
```
import java.applet.*;
import java.awt.*;

/* <applet code="Shapes.class" width=800 height-800> </applet>*/

public class Shapes extends Applet {
public void paint(Graphics g) {
setForefround(Color.red);
setBackGround(Color.blue);
//drawing squares
g.drawLine(10,10,100,10);
g.drawLine(10,10,10,10);
g.drawLine(10,100,100,100);
g.drawLine(100,100,100,10);
// Drawing triangle
g.drawLine(10,120,100,120);
```

```
g.drawLine(10,120,50,200);
g.drawLine(50,200,100,120);
//drawing Rectangle
g.drawRect(120,10,220,120);
g.fillRect(120,120,220,120);
//drawing ellipse and circle
g.drawOval(10,220,100,220);
g.setColor(Cplor.yellow);
g.fillOval(120,250,250,250);
//draw a filled arc
g.fillArc(350,50,400,100,0,90);
//draw a polygon
int x[]={400,500,400,500);
int y[]={240,240,340,340};
g.drawPolygon(x,y,4);
} }
```

Output:



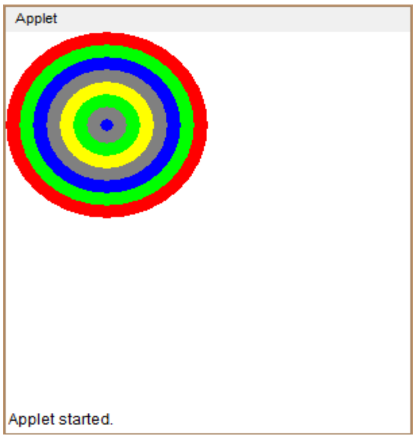
48. Applet Program for displaying circles

```
import java.awt.event.*;
import java.applet.*;
import java.awt.*;
/* <applet code="circle2.class" width=800 height-800> </applet>*/

public class circle2 extends Applet
{
public void paint(Graphics g)
{
Color
colors[]={Color.red,Color.green,Color.blue,Color.gray,Color.yellow,Color
.green,
Color.gray,Color.blue,Color.red,Color.pink};

int x=(getSize().width)/2;
int y=(getSize().height)/2;
for(inti=0;i<5;i++)
{
for(i=0;i<=colors.length;i++)
{
int j=i*10;
int k=i*20;
g.setColor(colors[i]);
g.fillOval(x+j-150,y+j-150,150-k,150-k);
}
}
g.drawString("width"+getSize().width,10,10);
g.drawString("height"+getSize().height,70,10);
}
}
```

Output:



49. Applet Program for bouncing balls

```
import java.applet.*;
import java.awt.*;

public class BouncingBall extends Applet implements Runnable
{
        // x,y coordinates and radius of the circle.
        int x = 150, y = 50, r=20;
        int dx = 11, dy = 7;

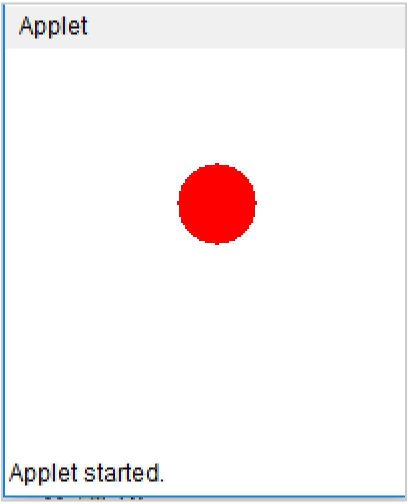
        // create thread.
        Thread t;
        boolean stopFlag;

        // Function to start thread.
        public void start()
        {
                t = new Thread(this);
                stopFlag=false;
                t.start();
        }

        // Draw cicle from its present position.
        public void paint(Graphics g)
        {
                g.setColor(Color.red);
                g.fillOval(x-r, y-r, r*2, r*2);
        }
```

```
// function to move the image.
public void run()
{
while(true)
{
if(stopFlag)
break;
// Bounce if we've hit an edge.
if ((x - r + dx < 0) || (x + r + dx > bounds().width)) dx = -dx;
if ((y - r + dy < 0) || (y + r + dy > bounds().height)) dy = -dy;
// Move the circle.
x += dx;  y += dy;
try
{
Thread.sleep(100);
}
catch(Exception e)
{
System.out.println(e);
};
// print circle again n again.
repaint();
}
}
// function to stop printing.
public void stop()
{
stopFlag=true;
t=null;
}}
```

Output



50. Move the text using applet.

```
import java.awt.*;
import java.applet.*;

public class MovingContent extends Applet implements Runnable
{
        // enter message
        String msg = "Welcome to Alagappa University";
        Thread t = null;

        // initialize here.
        int state;
        boolean stopFlag;

        // Set colors and initialize text..
        public void init()
        {
                setBackground(Color.cyan);
                setForeground(Color.red);
        }

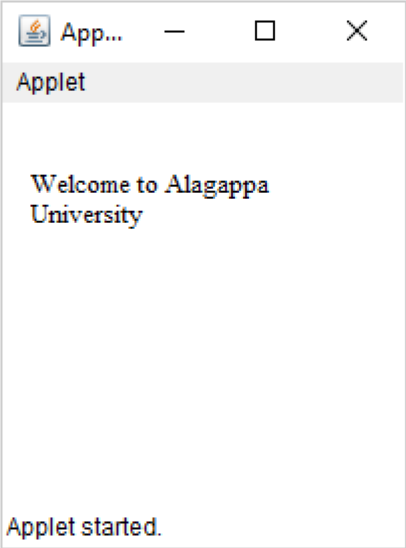
        // Start the text....
        public void start()
        {
        t = new Thread(this);
                stopFlag = false;
                t.start();
        }
```

```
        // Entry point which runs the text.
        public void run()
        {
        char ch;
        // Display text reapeated times.
        for( ; ; )
        {
        try
        {
        repaint();
        Thread.sleep(250);
        ch = msg.charAt(0);
        msg = msg.substring(1, msg.length());
        msg += ch;
        if(stopFlag)
        break;
        }
        catch(InterruptedException e)
        {
        System.out.println(e);
        }
        }
        }

        // Pause the text.
        public void stop()
        {
        stopFlag = true;
        t = null;
        }

        // Display the text.
        public void paint(Graphics g)
        {
                g.drawString(msg, 50, 30);
        }
}
```

Output



6.2 Network Programms

Java Networking is a concept of connecting two or more computing devices together so that we can share resources.

The widely used java networking terminologies are given below:

1. IP Address
2. Protocol
3. Port Number
4. MAC Address
5. Connection-oriented and connection-less protocol
6. Socket

1) IP Address

IP address is a unique number assigned to a node of a network e.g. 192.168.0.1. It is composed of octets that range from 0 to 255.

It is a logical address that can be changed.

2) Protocol

A protocol is a set of rules basically that is followed for communication. For example:

- TCP
- FTP

- Telnet
- SMTP
- POP etc.

3) Port Number

The port number is used to uniquely identify different applications. It acts as a communication endpoint between applications.

The port number is associated with the IP address for communication between two applications.

4) MAC Address

MAC (Media Access Control) Address is a unique identifier of NIC (Network Interface Controller). A network node can have multiple NIC but each with unique MAC.

5) Connection-oriented and connection-less protocol

In connection-oriented protocol, acknowledgement is sent by the receiver. So it is reliable but slow. The example of connection-oriented protocol is TCP.

But, in connection-less protocol, acknowledgement is not sent by the receiver. So it is not reliable but fast. The example of connection-less protocol is UDP.

6) Socket

A socket is an endpoint between two way communications.

The java.net package provides many classes to deal with networking applications in Java.

51. Program to display IP address of the given website

```
import java.net.*;
import java.io.*;
public class ip
{
public static void main ( String[] args ) throws IOException
{
```

```
String hostname = args[0];
try
{
    InetAddress ipaddress = InetAddress.getByName(hostname);
    System.out.println("IP address: " + ipaddress.getHostAddress());
}
catch ( UnknownHostException e )
   {
    System.out.println("Could not find IP address for: " + hostname);
   }
}}
```

Output:

```
C:\jdk1.3\bin>javac ip.java
C:\jdk1.3\bin> java ip www.google.com

IP address: 136.206.217.25
```

52. Program to display IP address of the localhost

```
import java.net.InetAddress;
import java.net.UnknownHostException;

public class Demo {
public static void main(String[] args) {
    InetAddress ipadd;
    String hostname;
    try {
      ipadd = InetAddress.getLocalHost();
      hostname = ipadd.getHostName();
      System.out.println("Your IP address : " + ipadd);
      System.out.println("Your Hostname : " + hostname);
      } catch (UnknownHostException e) {     }
}
}
```

Output:

```
C:\jdk1.3\bin>javac Demo.java
C:\jdk1.3\bin> java Demo
Your IP address : 4d623edc62d4/172.17.0.2
Your Hostname : 4d623edc62d4
```

53. Program to display parts of URL address

```
import java.net.URL;
public class Main {
   public static void main(String[] args) throws Exception {
      URL url = new URL(args[0]);
      System.out.println("URL is " + url.toString());
      System.out.println("protocol is " + url.getProtocol());
      System.out.println("file name is " + url.getFile());
      System.out.println("host is " + url.getHost());
      System.out.println("path is " + url.getPath());
      System.out.println("port is " + url.getPort());
      System.out.println("default port is " + url.getDefaultPort());
   }
}
```

Output:

```
>javac Main.java
>java Main https://www.alagappauniversity.ac.in/page/219

The URL is https://www.alagappauniversity.ac.in/page/219
The file part is /page/219
host is www.alagappauniversity.ac.in
path is /page/219
port is -1
default port
```

----------Try it yourself----------

1. Write a JAVA program to paint like paint brush in applet.
2. Write a JAVA program to display analog clock using Applet
3. Write a JAVA program that display the x and y position of the cursor movement using Mouse.
4. Write a JAVA program that identifies key-up key-down event user entering text in a Applet

****

Sub. Code 31544/ 34044

Model Question Paper

DISTANCE EDUCATION

M.Sc. (IT) DEGREE EXAMINATION

Second Year-Fourth Semester

Internet and Java Programming Lab

Time: Three hours **Maximum: 75 marks**

One question should be given to each candidate by Lot system

1. (a) Write a Java Program to define a class, describe its constructor, overload the Constructors and instantiate its object

   (b) Write a Java program for using Graphics class
   - to display basic shapes and fill them
   - draw different items using basic shapes
   - set background and foreground colors

2. (a) Write a Java Program to display the Prime numbers in given range

   (b) Write a Java Program to demonstrate Mouse events using applet

3. (a) Write a Java Program to display alphabetical order for the given set of names

   (b) Write a program to implement the concept of Exception Handling by creating user defined exceptions.

4. (a) Write a Java Program for student grade calculation

   (b) Write a program to implement the concept of threading by implementing Runnable Interface

5. (a) Write a Java Program to display reverse order of the given set of numbers.

   (b) Write a Java program to implement the concept of importing classes from user defined package and creating packages.

6. (a) Write a Java program to practice using String class and its methods.

   (b) Write a Java Program to implement multilevel inheritance by applying various access controls to its data members and methods.

7. (a) Write a Java program to implement the concept of Exception Handling.

   (b) Write a program to demonstrate use of extending interfaces.

8. (a) Write a Java Program to define a class, define instance methods and overload them and use them for dynamic method invocation

   (b) Write a program to implement the concept of threading by extending Thread Class